As previously noted, the simple name 304 generally contains an alpha numeric string of characters which uniquely identify a resource which this object 302 represents within its home context. The home context is determined by the home of the object 305 which is obtained from a reference to the home object 310. When the resource manager 321 initially creates the object 302, a user or other process (e.g., a discover process, to be explained) identifies another object 302 within an object hierarchy under which this object 302 is to be created. The initial placement location within an object hierarchy determines which object 302 within a hierarchy initially becomes a home object 310 for the object 302 and thus defines the home 305 of the object 302.

The resource manager 121 may require a representation of the object 302 in other locations within an object hierarchy. To provide for such other representations, parent object indicators 312 indicate the identity of other objects 302 besides the home object 310 under which a representation of this object 302 should be created in order to render the representation on the graphical user interface 150 with a resource manager 121 displays the graphical user interface 150. For each parent object indicator 312, a corresponding home condition 314 is included which the resource manager 121 uses to determine whether or not the home of the object 305 is to be included in the representation of the object when displayed within the graphical user interface 150.

The group object indicator 316 indicates whether or not the object 302 is to be considered a group object. As explained above, group object generally do not represent any one particular physical resource (i.e., do not represent a specific software or hardware resource), but rather, represent a collection or group of such resources which may be represented and acted upon collectively. The group object indicator 316 includes transparent and terminal indicators 318 and 320. Generally, the transparent and terminal indicators 318 and 320 contain, for example, boolean values that indicate whether or not this object 302 is to be considered transparent and/or terminal with respect to its placement within an object hierarchy in relation to other objects 302. As noted above, a transparent object does not act or serve as a home to objects placed below it in an object hierarchy. A terminal object, whose purposes will be explained in more detail later, generally prevents actions that are applied to the terminal group object from being

applied to that terminal group object's child objects. In other words, if a user attempts to apply an management function or action on an object that is terminal, that function or action will not be applied to objects related below that object in an object hierarchy.

The child object references 322 provide pointers, references or other indicators of any child objects which are related below this object 302 within an object hierarchy. The object properties fields 324 identifies any other properties related to a resource which this object 302 represents. As an example, the object properties field 324 may contain resource specific information such as device configuration information, software version or operation information, device or software serial numbers, or any other information which may be pertinent or beneficial to have in order for the object 302 to represent a specific resource within a computing system environment 100. In addition, the object methods 326 contain references to any functions, operations, methods, routines, libraries or procedures which defined logical operations related to the resource which this object 302 represents. Within a resource management application 120, the object methods 326 may define, for example, the management functionality or actions which a user 108 can apply or invoke to manage a specific resource associated with the object 302.

Figure 5 illustrates an operational environment 300 in which the resource manager 121 can receive input from the user 108 and optionally from a discover process 310 in order to access a resource pool 308 to produce an object hierarchy 301 containing objects 302 that represent various resources or groupings of resources in the resource pool 308. In this example, the object hierarchy 301 contains an object for each representation from the graphical user interface 150 in Figure 3. That is, there is a one to one correspondence between each object 302 in the object hierarchy 301 in Figure 3 and each representation of an object (e.g., 350 through 362) shown in the hierarchical graphical user interface 150 in Figure 3.

In Figure 5, the resource pool 308 represents a set of resources (e.g., 102 through 110 in Figure 1) within a computing system environment 100 which are accessible by the resource manager 121 for display and management within the graphical user interface 150. During the creation and representation of the objects 302 as previously explained with respect to Figure 2, the user 108 can use the resource manager 121 to query the

resource pool 308 of available manageable resources in order to create the objects 302 at proper locations within the object hierarchy 301. The resource manager 121 may also operate in conjunction with a discover process 310.

Generally, the discover process 310 is capable of automatically querying resources in the resource pool 308 (e.g., querying devices in a storage area network) to discover information about each resource and is then able to invoke the resource manager 121 according to the techniques as explained herein in order to create an object 302 that corresponds to each discovered resource and can further display a representation of that object on the graphical user interface 150. This process is repeated for each resource in the resource pool 308 resulting in an automated creation of the object hierarchy 301 and the graphical user interface 150 containing the representation 151 of the object hierarchy.

Returning attention back to Figure 3 and specifically to the graphical user interface 150, the example representation 151 of the object hierarchy contains a representation of each distinct object 302 in the object hierarchy 301 from Figure 5. That is, each object 302 is shown by a corresponding representation of an object in the graphical user interface 150. To display the graphical user interface 150, the resource manager 121 is able to traverse the object hierarchy 301 and, for each object 302 contained in the object hierarchy 301, is able to display a representation of that object 302 on the graphical user interface 150 according to the naming and grouping techniques explained herein.

The resource manager 121 displays most representations of objects from the object hierarchy 301 using only the simple name 304 of the object if the simple name uniquely identifies an object 302 in the context in which it is displayed within the graphical user interface 150. By way of example, in the graphical user interface 150 in Figure 3, the representations 350, 351 of the volume objects 302-13 and 302-22 from Figure 5 have the same simple name "VOL001" and do not need to include their respective homes 305 (e.g., "BIG BERTHA" and "LITTLE BERTHA,") since each is respectively displayed in its home context (i.e., under the representations of their home objects 302-28 and 302-29 (Figure 5).